Maceda, Rafael
2016-01337
CoE 197S

Feature Activity 05

A. Setting up the server

To run our Apache HTTPD server, run this command:

```
$ docker run --rm -d --name apache -p 80:80 httpd:2.4
d87e0a193dde5652ac762d8849983c2cadb5116b80c8a61a4180e350d678b4d2
```

This command will start a new container from HTTP 2.4, name it `apache`, bind port `80` to the host machine (more on this later), and set a flag to delete the container when it stops.

After it starts, we can run `curl localhost` to query the web server for the default page:

```
$ curl localhost
<html><body><h1>It works!</h1></body></html>
$
```

```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>docker run --rm -d --name apache -p 80:80 httpd:2.4
Unable to find image 'httpd:2.4' locally
2.4: Pulling from library/httpd
69692152171a: Pull complete
7284b4e0cc7b: Pull complete
3678b2d55ccd: Pull complete
aeb67982a725: Pull complete
06954f8169fd: Pull complete
Digest: sha256:48bae0ac5d0d75168f1c1282c0eb21b43302cb1b5c5dc9fa3b4a758ccfb36fe9
Status: Downloaded newer image for httpd:2.4
ec28b532cdc54655e423e62ad4df8989a74c5d88f09ce1ee8fc1ac3774187966

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>
```

```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>curl localhost
<html><body><h1>It works!</h1></body></html>
```

This is the default `index.html` file included with a new Apache 2.4 installation. Let's replace this HTML file with new content.

To do so, we'll use the `docker cp` command, similar to `scp`, which copies files between the host and containers. Let's give it the `index.html` file from the directory this README is sitting in:

```
$ docker cp index.html apache:/usr/local/apache2/htdocs/
$
```

```

The first path is the source path, representing our new file on our host machine, and the second path our destination. `apache` is the name of the container we want to copy into, and `/usr/local/apache2/htdocs/` is where the web server serves HTML from.

Running `curl` again now looks a little different:

```
$ curl localhost
<html><body><h1>It works in Docker!</h1></body></html>
$
```

```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker cp index.html apache:/usr/local/apache2/htdocs/

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>curl localhost
<html><body><h1>It works in Docker!</h1></body></html>
```

B. A possible data problem

This container, for its lifetime, will continue to serve our new HTML file.

However, containers in Docker are, in practice, considered ephemeral. They can die unexpectedly, and in certain deployments, be removed without warning. If you're depending upon the container state for your application, you might lose important data when such containers die. This is especially a concern for applications like databases, which are supposed to be considered permanent datastores.

In the case of our HTTPD server, simply stopping the container will cause it to be autoremoved. We can bring another container back up in it's place, but it won't have our changes any more.

```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker stop apache
apache

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker run --rm -d --name apache -p 80:80 httpd:2.4
d548b851b4d093153c13e7fc1d7ccd67747ea317c1c27b120d19b9e986518ef3

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>curl localhost
<html><body><h1>It works!</h1></body></html>

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>
```
```

C. Managing volumes

Volumes in Docker are file stores, which sit independently of your Docker containers. The function like Amazon Web Services' EBS volumes, and other mountable media like USB thumb drives. They can be created, deleted, and mounted on containers at specific locations within an image, like you would with `mount` command in Linux.

To list your volumes, run `docker volume ls`:

To create a new volume, run `docker volume create` and give it a volume name.

```
$ docker volume create myvolume
myvolume
$ docker volume ls
DRIVER              VOLUME NAME
local               myvolume
$
```

To remove a volume, run `docker volume rm` and give it the volume name.

```
$ docker volume rm myvolume
myvolume
$ docker volume ls
DRIVER              VOLUME NAME
$
```

```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker volume ls
DRIVER    VOLUME NAME

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker volume create myvolume
myvolume

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker volume ls
DRIVER    VOLUME NAME
local     myvolume

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker volume remove myvolume
myvolume

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker volume ls
DRIVER    VOLUME NAME
```

D. Mounting volumes on containers

First create a new volume named `httpd_htdocs`:

```
$ docker volume create httpd_htdocs
httpd_htdocs
$
```

Then re-run our `docker run` command, providing the `-v` mount flag.

```
$ docker run --rm -d --name apache -p 80:80 -v
httpd_htdocs:/usr/local/apache2/htdocs/ httpd:2.4
c21dd93fea83d710b4d4c954911862760030723df6a5b42650e462e388fe6
049
$
```

And re-copy in our modified HTML file.

```
$ docker cp index.html apache:/usr/local/apache2/htdocs/
$
```

And run `curl` to verify it worked.

```
$ curl localhost
<html><body><h1>It works in Docker!</h1></body></html>
$
```

Now to see the volume in action, let's stop the container. By providing the `--rm` flag during `run`, it should remove the container upon stopping.

```
$ docker stop apache
apache
$
```

Now to see the volume in action, let's stop the container. By providing the `--rm` flag during `run`, it should remove the container upon stopping.

```
$ docker stop apache
```

```
apache
$
```

Then once again start httpd with the same run command as last time. This time, however, we can `curl` and see our file changes are still there from before.

```
$ docker run --rm -d --name apache -p 80:80 -v
httpd_htdocs:/usr/local/apache2/htdocs/ httpd:2.4
c21dd93fea83d710b4d4c954911862760030723df6a5b42650e462e388fe6
049
$ curl localhost
<html><body><h1>It works in Docker!</h1></body></html>
$
```

```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker run --rm -d --name apache -p 80
d:2.4
ba28170b6634bf2208f80639a29bd1a0b68bcae5cafe525b6c137d7eb766d108

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>curl localhost
<html><body><h1>It works in Docker!</h1></body></html>
```

E. Mounting host directories on containers

As an alternative to using volumes, if you have a directory on your host machine you'd like to use like a volume, you can mount those too. This is technique is useful in development environments, where you might want to mount your local repo onto a Docker image, and actively modify the contents of a Docker container without rebuilding or copying files to it.

The `-v` flag to accomplish this is almost identical to the previous one. Simply specify an absolute path to a local directory instead. In our case, we'll pass `.` to specify the `5-volumes` directory in this repo, which conveniently contains a modified version of the HTML file.

```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\5-volumes>docker run --rm -d --name apache -p 80:8
on_and_Docker/5-volumes/:/usr/local/apache2/htdocs/ httpd:2.4
12aaedf755960885607b249c8735caaa23ccda1b2c6805abeb794a92a6e10744
```

With the host directory mount in place, modify the `index.html` file in this directory with whatever message you like, then save the file and re-run `curl`.

```
```

```
$ curl localhost
<html><body><h1>It works quite well in
Docker!</h1></body></html>
$
```

You can see file changes take place immediately on the Docker container without any need to run `docker cp`.

Go ahead and run `docker stop apache` to stop and remove the container.